BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Problem Image Mailbox.

A Test and Maintenance Controller for a Module Containing Testable Chips*

Melvin A. Breuer and Jung-Cheun Lien

University of Southern California
Department of EE-Systems
Los Angeles, CA 90089-0781

Abstract

A design of a module test and maintenance controller (MMC) is presented in this paper. Driven by structured test programs, an MMC is able to test every chip in a module via an ETM-BUS or a Boundary Scan bus. More than one test bus can be controlled by an MMC. MMC processor instructions, when executed, produce bus timing sequences which control a chip's BIT structures. The proposed MMC is a universal design. The difference between MMCs on different modules is the test programs which they executed and the number of test busses they control. Performance analysis indicates that either a RISC-type processor or DMA controller is required in the MMC. Some self-test features of the MMC are also presented.

1 Introduction

Designing testable chips that are compatible with standard test busses has recently drawn much attention [1,2,3,4,10,11,12]. Two major initiatives have emerged over the last few years. One is the ETM-BUS protocol proposed by the VHSIC committee [5]; another is the Boundary Scan protocol proposed by the JTAG committee [7]. In addition to these two promising standards, an IEEE subcommittee is proposing a T-BUS standard which is intended to accommodate all major test busses [13]. The main objective of these efforts is to support design for testability (DFT) of a module (or a board). An acceptable degree of testability is not always achievable by simply using a set of testable chips unless they are properly integrated at the module level.

*This work was supported by Defense Advanced research Projects Agency and monitored by the Office of Naval Research under contract no. N00014-87-K-0861.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

In designing a Hierarchically Testable and Maintainable (HTM) system [18], which has a very high degree of testability and maintainability, an on-module test and maintenance controller (MMC) is required in the test control hierarchy. The same concept can be found in a system with multilevel self-test capability [19].

An MMC is able to control the self-test process of a module containing testable chips by accessing each chip's BIT structures through either an ETM-BUS or a Boundary Scan bus. The proposed MMC is universal in that the same basic design is used for all modules. The only difference between different MMCs is the test programs they execute and the number of test busses they control. These test programs can direct the processor in an MMC to execute a BIST process for the entire module. The test results are then reported to a subsystem test and maintenance processor (SuMP) via a TM-BUS [6]. A SuMP can initiate the self test process of a module beneath it by sending a "begin test" command to the MMC on that module. For further details about a SuMP the reader is referred to [18].

Figure 1 shows part of the test hierarchy associated with a module containing two different test busses. An MMC contains bus interface units, such as a TM-slave and ETM-master, a processing unit such as a processor, a memory unit consisting of RAMs and ROMs, one or more test channels, an on-chip test and maintenance controller (CMC), and a data transfer control unit such as DMA controller. Only bus interfacing units are shown in the figure.

A TM-slave serves as a medium between an MMC and a SuMP. It executes the TM-slave protocol as well as provides temporary storage for information received or to be sent over the TM-BUS. No information can be transmitted between the TM-slave and the MMC processor while the TM-slave and the SuMP are transmitting information and vice verse. When the TM-slave has received information from a SuMP, an internal MMC interrupt is generated to request service. The MMC processor or DMA controller is then used to transfer this information from the TM-slave to the memory unit of the MMC.

1988 International Test Conference Proceedings - New Frontiers in Testing
12-14 Sep 1988

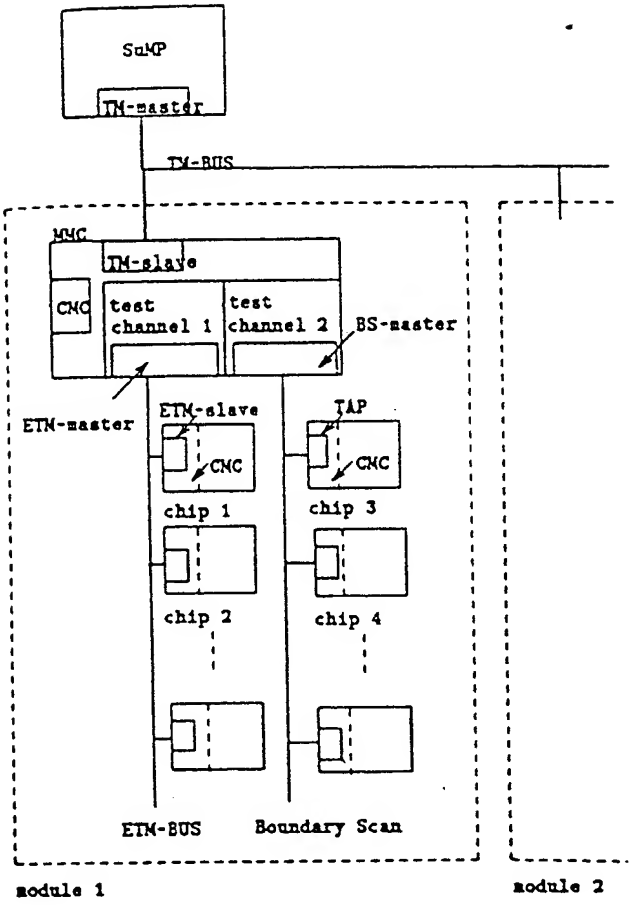


Figure 1: Test hierarchy for a module.

A simple yet novel design, called a test channel, is used in an MMC. It is assumed that all testable chips have an on-chip test and maintenance controller (CMC) which contains an ETM-slave when an ETM-BUS is used, or a Test Access Port (TAP) when a Boundary Scan bus is used. A test channel, which contains an ETM-master, can communicate over an ETM-BUS with a CMC through its ETM-slave. The processor can control a test channel by reading or writing its internal registers. Once initiated by the processor, a test channel can completely control an ETM-BUS and the testing of a chip. Information flow between a test channel and the memory unit can be either controlled by the processor, which is usually slow, or by a DMA controller, which is much faster.

The separation of processor and test busses provided by test channels prevents the processor from dealing with detailed bus timing activities. A test channel can translate processor instructions into proper timing sequences for a test bus. Execution of a test process now can be represented as high level processor instructions.

In section 2 a generic test control model for a testable chip is presented. Two popular testable design techniques and associated test plans are reviewed. A design of an MMC is presented in section 3. The requirements for an MMC are described first, followed by its architecture. Major building blocks such as a test channel, TM-slave, processor and memory are described in turn. Some self-test aspects of the MMC are also presented. Analysis for the data transfer requirements between memory and test channels is given in section 4.

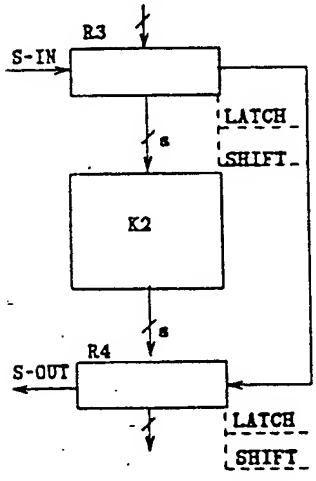
2 Test Control Models

2.1 Testable Design Techniques

Systematic techniques are used extensively in designing testable chips [14,15,16,17]. Two commonly used testable design techniques, namely LSSD[16] and BILBO[17], are described in the following. Their associated test plans for a kernel are also presented.

The LSSD technique requires all storage cells in a CUT to be made into a shift register latch (SRL) which are then interconnected into one or more shift registers or scan chains. Two data inputs, viz. functional data and shift (test) data, gated by separate clocks C and A, are used to load data into a SRL. Clocks A and B are used for shifting data in the scan chains, while clocks C and B are the functional clocks. In addition to the functional design of a CUT, only 4 additional I/O lines are required to test the CUT, viz. Scan In, Scan Out, Clock A and Clock B.

In Figure 2(a) a CUT with one kernel is shown. The scan chain consists of 2s SRLs. Assume this kernel requires t vectors to test.




```
Test Schema LSSD
Head
   do s times R3(SHIFT)
   propagate through K2
   R4(LATCH)
Body
   do t-1 times
      do s times
         R3(SHIFT)
         R4(SHIFT)
      propagate through K2
      R4(LATCH)
Tail
   do s times R4(SHIFT)
```

(a) (b)

Figure 2: An LSSD kernel.

(a) Test structure;
(b) Test schema.

The test plan for this CUT is as follows. (1) Shift in a test vectors (by activating clocks A and B s times); (2) issue a single step operation for the CUT (by activating clocks C and B once); (3) shift out the test result for the previous vector and shift in a new test vector (by activating clocks A and B s times); (4) repeat steps 2 & 3 (t-1) more times; (5) shift out the final test result (by activating clocks A and B s times); and (6) compare all test results with precomputed correct ones.

To control such a test plan, the controller must be able to provide proper clock switching between A, B and C, to count to s and t separately, to enable a single step operation, to provide test vectors and to process test results.

The BILBO technique is used to create a Built-In Self-Test (BIST) circuit. All registers in a CUT are converted to BILBO registers. A BILBO register, controlled by two lines B1 and B2, can operate in any one of four modes, namely LATCH, RESET, SHIFT and PSA. In the SHIFT mode, BILBO registers act as shift registers, and seed and other control patterns can be loaded into them and signatures can be scanned out. In the PSA mode, all BILBO registers act as parallel signature analyzers.

Two BILBO registers are required to test a kernel. One acts as a Test Pattern Generator (TPG) and the other as a PSA during BIST operation. A PSA can act as a TPG if all its parallel data inputs are held constant. So extra control signals must be provided to decide whether a BILBO should act as a PSA or a TPG during the PSA mode. In [10] an extra bit is put in the BILBO design for this purpose.

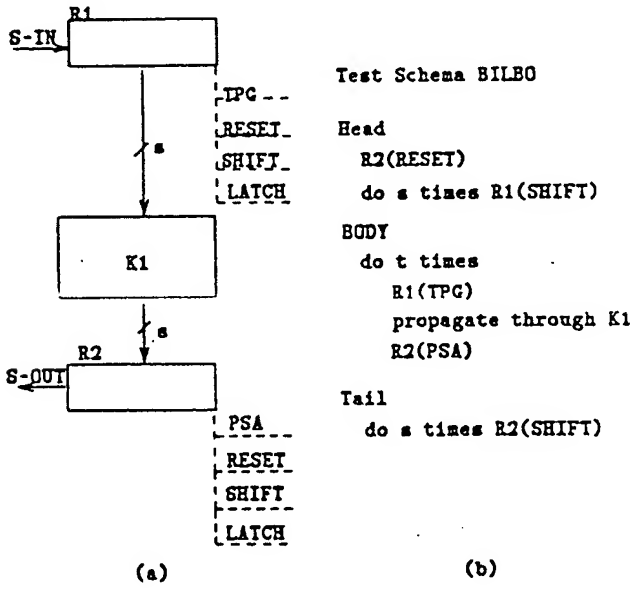


Figure 3: A BILBO kernel.

(a) Test structure;
(b) Test schema.

An embedded BILBO structures is shown in Figure 3. Here K1 is the circuit to be tested and R1 and R2 are both BILBO registers. R1 acts as a TPG and R2 as a PSA. Only two control lines B1 and B2 are required to control all BILBO registers.

The test plan for this structure is as follows. (1) (SHIFT mode) Shift in seeds and control bits for R1 and R2; (2) (PSA mode) put both R1 and R2 into the PSA mode and run t functional clocks; (3) (SHIFT mode) shift out result (signature) in R2; and (4) compare the signature with the precomputed correct one.

To control such a test plan, the controller must be able to provide control signals to B1 and B2 for the BILBOs, to provide seed data and control bits to distinguish between TPG and PSA mode, to count to s and to t, and to process test results.

From the above discussions we can conclude that in order to control both LSSD and BILBO structures, a test controller must be able to provide data to the CUT (either test vectors or seeds), to switch between test and functional clocks, to provide required control signals, to count to s and t and to process test results. More details on DFT and BIST test controllers can be found in [8,9].

A generic test control model for a testable chip is presented next. The control and data required to test a chip is supplied by the MMC which communicate with the CUT via a test bus. Based on such a model the architecture of a MMC is then proposed.

2.2 A Generic Model for Testable Chips

It is necessary to derive a generic test control model for all testable chips. In this model, a testable chip is assumed to be accessed via an ETM-BUS or a Boundary Scan bus. Thus a bus interface, such as an ETM-slave or a TAP, must be part of the testable chip. The control process using an ETM-BUS or Boundary Scan bus are very similar. For the rest of this paper we will focus primarily on the ETM-BUS.

The MMC transmit two types of information to a chip, namely instructions and data. Instructions are sent to the instruction register in an ETM-slave and control the test function of a chip, while data is sent to a selected scan chain in the chip. Two types of information are sent from the chip to the MMC, namely status and results. The status is the contents of the status register in an ETM-slave and indicates the current status of the chip, while results come from a selected scan chain in the chip.

Figure 4 shows a generic on-chip test control model for a typical testable chip consisting of n scan paths. The generic test control model has the following attributes.

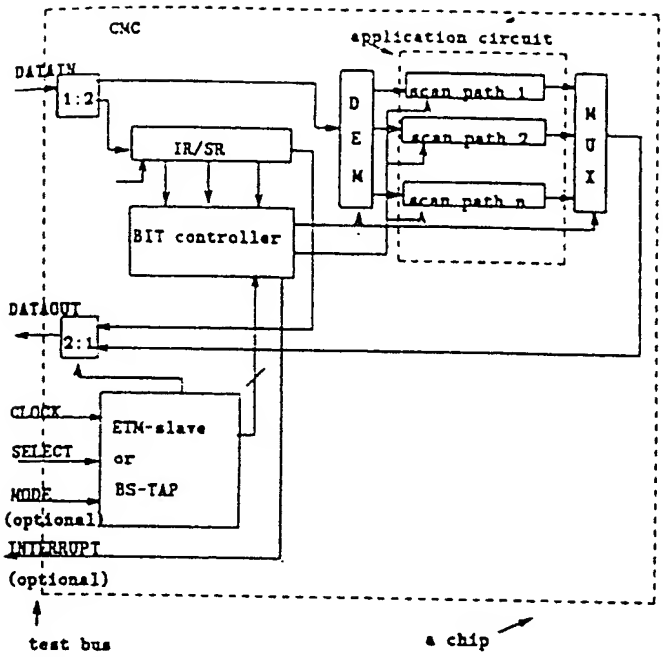


Figure 4: Control model for a testable chip.

1. A dedicated test bus, such as the ETM-BUS or Boundary Scan. The signals for an ETM-BUS (Boundary Scan) are as following, (1) DATAIN: (TDI) for serial test data input. (2) DATAOUT: (TDO) for serial test data output. (3) REFCLK: (TCK) a dedicate test clock. (4) SELECT: (TMC) used to enable the exchange of information between an ETM-master and several ETM-slaves. (5) MODE: used to indicate either Instruction or Data exchange is in progress. (6) INTERRUPT: a status indicator used to send a signal from ETM-slaves to the ETM-master.

2. A bus interface to accommodate the above 6 (or 4) signals. The bus interface acts as either an ETM-slave or a TAP. Through this interface, an MMC is able to control the chip's BIT structures.

3. A centralized control scheme is assumed. This means that the test bus is tied up during the entire test process. The MODE and SELECT lines drive a FSM in the CMC which, along with the instruction register, produces control signals for the BIT structures.

4. A testable chip can have one or more scan chains. For chips employing the BILBO technique, this model represents the process of initial seeding, test configuration and signature output.

5. Two different operation modes exist for an ETM-slave, namely instruction/status shift mode and data/result shift mode. The test bus signal MODE determines the operation mode.

6. For instruction/status mode, information shifts through an internal instruction/status register, while the target chip's functional operation may or may not be affected. The instruction register controls the operation of BIT structures on the chip. The status register contains the current status of BIT structures and/or the ETM-slave itself.

7. For data/result mode, test data and test results move along one of the internal scan chains. The target chip is usually in an off-line mode during this operation.

8. The DATAOUT of one chip can feed the DATAIN of another chip. Thus a long scan chain or a ring structure can be formed.

9. The content of the instruction register controls the test operation of a chip. Semantics for an instruction depends on the actual implementation of each chip. However, some basic instructions have been proposed by the VHSIC and JTAG committees [5,7].

3 MMC Design

A hierarchical test and maintenance (HTM) system has four level of controllers, viz. a system test and maintenance processor (SMP), subsystem test and maintenance processors (SuMP), module test and maintenance controllers (MMC), and on-chip test and maintenance controllers (CMC). An MMC must be able to respond to requests from a SuMP, to carry out tests for every chip on the module, and to report test results to the SuMP. The requirements for an MMC are stated in the next section, followed by a description of its architecture.

3.1 Requirements for an MMC

Based on the test control model presented, one can develop an MMC to satisfy all requirements for testing a module containing testable chips. An MMC should be able to support the following functions.

1. Access the on-chip BIT structures by using the generic test control model.

2. Provide proper control sequences and execute test plans for a chip's BIT structures.

3. Provide test data and collect test results if necessary.

4. Analyze test results to decide on the health status of chips.

5. Check the I/O among different chips on the module via Boundary Scan registers.

6. Interface with the system test control hierarchy.

Obviously, an MMC should have memory to store test data and/or test results if deterministic test data is used. For

random or exhaustive test methodologies, much less memory is required since only seeds and signatures need to be stored.

An MMC communicates with chips via a test channel, which is able to send instructions and three different types of test data, viz. deterministic, exhaustive and pseudorandom.

Also an MMC should have a TM-slave (or equivalent) which is controlled by a SuMP through a TM-BUS. The TM-slave receives instructions from a SuMP and the MMC's processor executes these instructions. Test results collected by an MMC can be reported to a SuMP through the TM-slave.

3.2 MMC Architecture

Figure 5 shows the architecture of an MMC. It consists of a TM-slave, either a 16 bit general or special purpose processor, a ROM, a RAM, one or more test channels, a DMA controller and a CMC which contains an ETM-slave and is used to control the self-test of the MMC. Only one CMC is required if the MMC is integrated into a single chip.

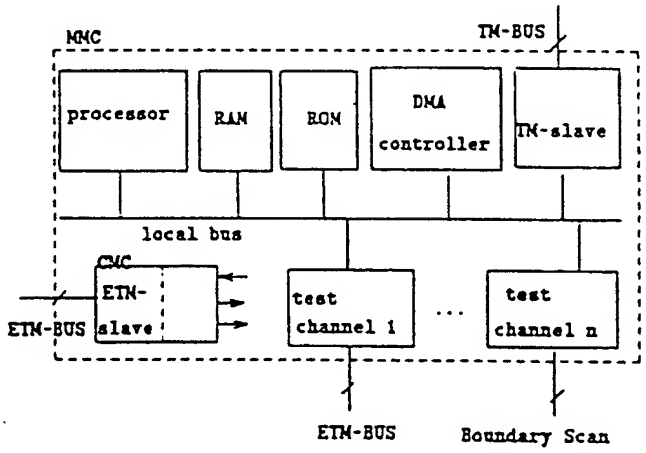


Figure 5: The architecture of an MMC.

For the current design, an MMC is composed of several chips. Note that every testable chip must have a CMC. so more than one CMC may exist, though only one is shown in the figure.

3.2.1 Test Channel Design

A CMC may have a TPG and an SA as part of the ETM-slave. In this case only control signals need be supplied by a test bus during self-test. An example of such a design for an ETM-slave is presented in [11]. However, if the chip does not have these registers and is to be tested using a pseudorandom test methodology, then a TPG and an SA must be made a part of the MMC. For chips use deterministic test vectors, an MMC must be able to provide test vectors and obtain test results through a test channel.

The major functions of a test channel are listed below.

1. Serve as an ETM-master or a Boundary Scan master (BS-master).
2. Provide a TPG and a SA for pseudorandom testing.
3. Transmit pseudorandom data to and receive test results from chips.
4. Transmit instructions to and receive status from chips.
5. Transmit deterministic test vectors to and receive test results from chips.
6. Generate interrupts and also direct interrupts from chips to the processor.
7. Provide DMA interface circuit to enable high speed information exchange with memory units.
8. Provide internal registers selection logic for processor interfacing.

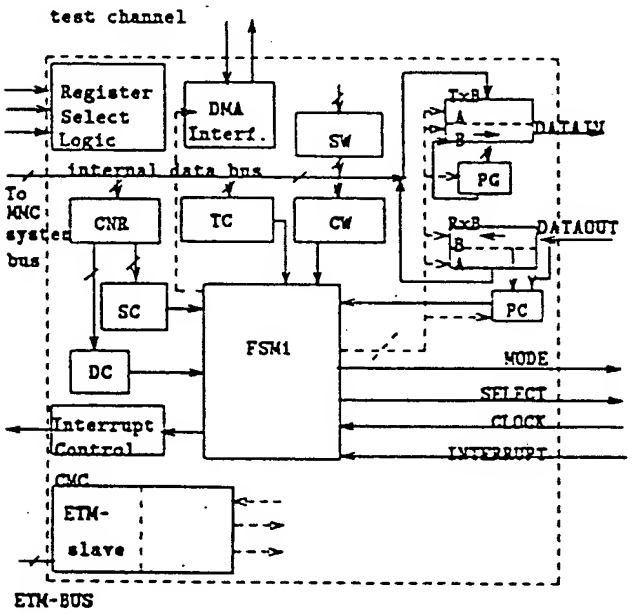


Figure 6: The architecture of a test channel.

Organization of a Test Channel: Figure 6 shows a block diagram of a test channel. It consists of a Transmitter Buffer (TxB) for transmitting data on the DATAIN line; a Receiver Buffer (RxB) for receiving data on the DATAOUT line; a parity generator (PG) to generate odd parity for instructions transmission; a parity checker (PC) to check odd parity for status transmission; a control word (CW) contains operation modes and function enabling information; a status word (SW) stores the current chip status; three counters, one (TC) keeps count of the total number of test vectors to send, the other (SC) counts number of shifts for each vector and another (DC) keeps count of the delay

of clocks between two vectors; a register (CNR) to restore values for SC and DC; a register select logic for processor read/write control; an interrupt circuit to request service from processor; a control unit (FSM1) which implements the ETM-master protocol is used to send and receive information via the ETM-BUS under the control of the CW and the three counters; a CMC containing an ETM-slave used for the self-test of the test channel; and a DMA interface for high speed information exchange with the MMC memory. A more detailed description for the major blocks follows.

1. TxB (Transmitter Buffer). The TxB consists of two 16 bit registers. It is used to transmit data on the DATAIN line. Buffer A has parallel LOAD capability, and can load data from the internal data bus. Buffer B has parallel LOAD, SHIFT and TPG capabilities.

   A write operation from the processor or DMA controller loads data in parallel from the internal data bus into buffer A. The data then can be copied into buffer B in order to be shifted out on the DATAIN line.

   During instruction or deterministic data transmission, data on the DATAIN line is shifted out from buffer B. As soon as buffer B is empty, the content of buffer A is copied into buffer B. At the same time a write request is generated and the processor or DMA controller must write buffer A before buffer B is again empty in order to prevent any interrupt of data on the DATAIN line.

   During pseudorandom data transmission, buffer B acts as a TPG. The seed for TPG is first written into buffer A by the processor. The content of buffer A is then copied into buffer B before test data is generated.

2. RxB (Receiver Buffer). The RxB is also a double buffer consisting of two 16 bit registers. It is used to receive data from the DATAOUT line. Buffer A has parallel LOAD and tri-state output capabilities, and buffer B has SHIFT and SA capabilities. Buffer B is not directly accessible by the processor. A read operation to the RxB always reads out the content buffer A.

   During status or deterministic result transmission, data on the DATAOUT line is shift into buffer B serially. As soon as buffer B is full, it is copied into buffer A automatically so that new data can be shifted into buffer B. Whenever buffer A is full, a read request is generated. To prevent any loss of data, the processor or DMA controller must read buffer A before buffer B is full.

   During pseudorandom data transmission, Buffer B acts as a SA. All incoming data will be compressed into the SA. Once finished, the signature in buffer B is copied into buffer A, and a processor read is requested to read out the signature from buffer A.

3. CW (Control Word). It consists of 3 registers. A 1-bit Start register which, when set will initiate the operation of FSM1 in the test channel. A 5-bit Mask Register (MR) is used to control the enabling and disabling of selected functions of the test channel. A 16-bit Configuration Register (CFR) controls the configuration and selection of chips to be tested. Two processor write operations are required to load the CW.

4. SW (Status Word). It consists of an 8-bit register with tri-state output capability. The current status of the test channel is stored in this word. Each bit can be set individually. It can be read out in parallel by a processor read operation. Some bits in the SW will be cleared after a read operation.

5. TC (Test Counter). It is used to count the total number of test vectors to be applied in one test session. The TC is a 22-bit down counter and requires two processor write operations to load. One of the write operations loads part of this counter and part of the CW. This counter is able to count down to 0 from any number between 1 and 4,194,303. When reset, the content of the counter will be set to its maximum value 4,194,303.

6. SC (Scan Counter) It is used to count the number of shifts for a test vector or an instruction. The SC is a 10-bit down counter and can count down to 0 from any number from 1 to 1023. Its initial value is loaded from the CNR. A terminal count signal will be activated whenever the value in SC reaches 0, and the value in CNR will be restored. When reset, its content will be set to its maximum value 1023.

7. DC (Delay Counter). DC is a 6-bit down counter and is used to count the number of clock cycles between the transmission of test data. Its initial value can be loaded from the CNR. The DC can count down to 0 from any number from 1 to 63. A terminal count signal will be activated whenever DC reaches 0, and the value in CNR will be restored. When reset, its content will be set to 63.

8. CNR (Count Number Register). This buffer is used to store the initial value of the constants for both SC and DC. These counters destroy their original contents after a test vector is transmitted. Thus this register is used to restore the value of both SC and DC so the next vector can be transmitted. The CNR is 16 bits long. It can be loaded by a processor write operation.

9. Register Select Logic. This circuit enables the test channel to interface with any general purpose pro-

cessor. It is used to select an internal register for a processor write or read operation.

10. DMA interface. This circuit provides the necessary DMA handshaking signals to a DMA controller. Its main purpose is to transfer data between memory and TxB (or RxB) at a high bandwidth. The user can choose either DMA or processor polling operation for data transfer. In DMA mode the chip will be activated via a so called soft select without actually asserting the Chip Select (CS) signal. All required signals are provided by the DMA controller. During DMA operation, the read signal is provided by IOR and the write signal by IOW, both generated by the DMA controller.

11. CMC. In this design a test channel is assumed to be a chip. To make a test channel testable a CMC which contains an ETM-slave is required.

12. FSM1. This circuit controls the operation of a test channel. Controlled by CW and terminal count signals from counters, the FSM1 acts as an ETM-master. It provides all control signals for the ETM-BUS and data to/from the bus. The FSM1 can also act as a BS-master. By setting a bit in the CW, the FSM1 is able to send and receive information to/from a Boundary Scan bus.

Operation of the Test Channel: The test channel can transfer three types of information, viz. Pseudorandom Testing Data (PTD), Deterministic Test Data (DTD) and Instructions (INS). The operations of the test channel are controlled by the CW and the terminal count signals of the three counters. These counters are used for all three types of information transfer. During different operations, a counter may be used for different purpose. The following table shows the usage of these counters.

	PTD	DTD	INS
TC	no. of test vectors	no. of test vectors	set to 1
SC	no. of shifts	no. of shifts	no. of shifts
DC	no. of delays	set to 16	set to 17

After all internal registers of a test channel have been initialized, setting the Start bit and enabling bit MR[2] in the CW will initiate the operation of the FSM1. Based upon a flag bit in the CW, the FSM1 can act as either an ETM-master or a BS-master. Figure 7 shows part of the state diagram for the FSM1. Three major branches exist for the transfer of different types of information. The branch labeled PTD is followed when pseudorandom testing is used. The branch labeled DTD is used when deterministic test data is employed. For transmitting instructions, the procedure is very similar to that of the DTD transmission. The only difference is that signal MODE is set to 0. Also note that the content of TC is always set to 1 for instruction transmission, while for DTD and PTD transmissions, the TC is set to the number of test vectors to be applied. The looping conditions depend on three terminal count signals (tc.TC, tc.SC, tc.DC) generated by counters TC, SC and DC, respectively. The processor can stop or disable the operation of the FSM1 by loading a new word into the CW through a processor write operation. Resetting the FSM enabling bit in CW (MR[2]) will halt the operation of the FSM1. In order to maintain consistent operation, modification of all other registers by the processor is prohibited until the FINISH bit in the SW is set or an interrupt has occurred.

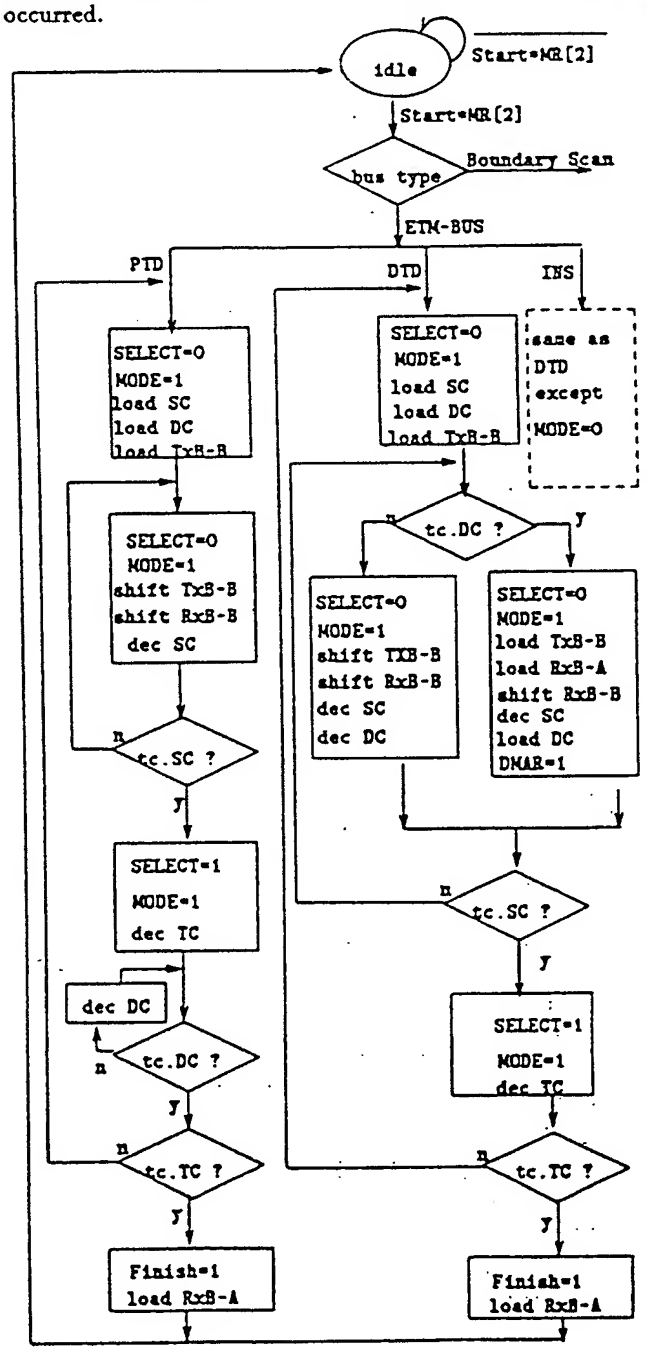


Figure 7: The state transition diagram for a test channel.

3.2.2 TM-slave Design

The TM-BUS protocol: A SuMP is able to communicate with one or more MMCs via a TM-BUS which consists of four lines, viz. CLOCK, CONTROL, MASTER DATA and SLAVE DATA. Information is sent from the master to slaves over the CONTROL and MASTER DATA lines, and in the opposite direction over the SLAVE DATA line.

The SuMP must have a TM-master to control bus activities, while each MMC must have a TM-slave to respond to requests from the bus. All activities are initiated by the TM-master except for interrupt.

Information transferred from master to slaves is in packets, consisting of 16 bits plus a parity bit. There are three types of packet, namely HEADER, ACKNOWLEDGE and DATA. A HEADER packet contains the address of a destination slave (8 bit) along with an instruction (7 bit) and an acknowledge bit plus a parity bit. The address of a slave consists of two parts, viz. a module identifier (5 bit) and a submodule identifier (3 bit). Up to 31 module can be connected to a TM-BUS, each containing at most 8 submodules addressable by the master. The highest module identifier (11111) plus its submodule identifiers are reserved for broadcast and group multicast operations.

Depending on the instruction in the HEADER packet, either one or more DATA packets are transmitted between master and slave. The master can temporarily interrupt the sending of a DATA packet by deactivating the CONTROL line. When the CONTROL line is inactive, any number of slaves can request service via an interrupt by activating the SLAVE-DATA line. Since all SLAVE-DATA lines are wired-ored together, the master must initiate a contend sequence in order to identify that slave with the highest priority (address) which has requested service.

With this protocol it is possible to transmit commands, test programs and test data from a SuMP to an MMC. In addition test results can be sent back from the MMC to the SuMP.

The organization: The TM-slave is designed as a peripheral device to the processor on an MMC. Information exchange between TM-slave and the processor, which is achieved by a processor read or write operation, is in parallel. Figure 8 shows the block diagram of a TM-slave. It consists of an input shift register (inSR) to accept data from MASTER DATA (MD) line; an output shift register (outSR) to provide data to the SLAVE DATA (SD) line; an Address Matching circuit to determine if this slave is selected; an Instruction Register (IR) containing the current instruction for the slave; a Status Register (SR) containing the current status of the slave; an Interrupt Logic network to generate an interrupt request on the SD line; an Input Buffer to store DATA packets received from the MD line; a counter (Counter-0) to indicate the number of bits still to be received for the current packet; a counter (Counter-1) to indicate how many packets were received over the TM-BUS; an Output Buffer to store DATA packets to be sent over the SD line; a counter (Counter-2) to indicate how many packets are to be sent; a Finite State Machine (FSM2) to control the operation of the TM-slave; a Processor Read/Write Interface to allow parallel data transfer between memory and internal registers of the TM-slave; a Processor Interrupt Circuit to request service from the processor; a DMA Interface Circuit (optional) to allow fast data transfer between the Input (Output) Buffer and the MMC memory; and a CMC which contains an ETM-slave for the self-test of the TM-slave.

Some units in the TM-slave are described in more detail below.

1. Address Matching circuit: This is a combinational circuit used to compare the module identifier (MID) with the received address in a HEADER packet. If a match occurs, it sends an enable signal to FSM2 which then allows further operations on the bus. When broadcast and multicast addresses are received, this circuit also generates an enable signal.

2. Input Buffer (Output Buffer): According to the TM-BUS protocol, up to 256 DATA packets can be received and/or transmitted in one transaction. Thus each buffer must have 256 16-bit words. These two

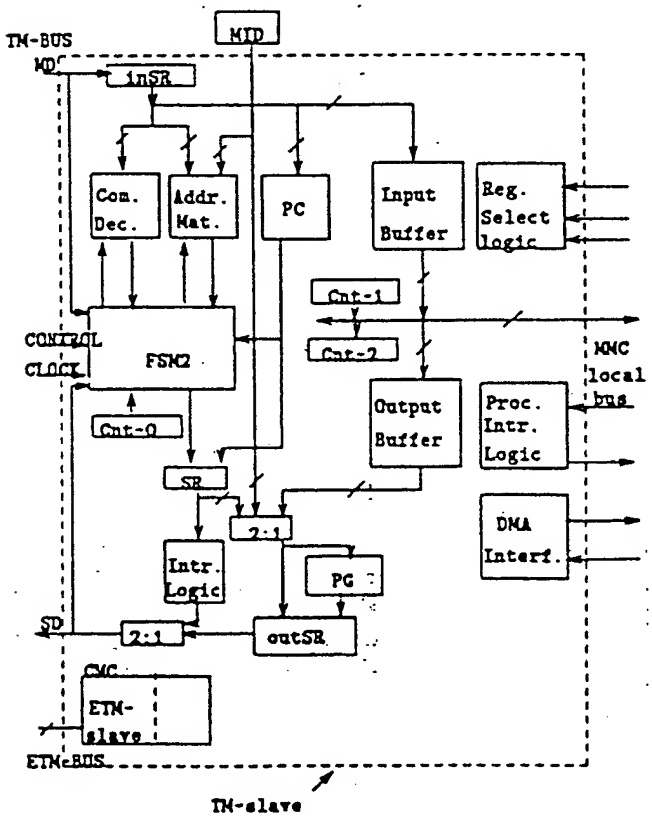


Figure 8: The architecture of a TM-slave.

buffers can be combined into a FIFO (First In First Out) buffer which allows simultaneously input and output. The buffer receives (sends) DATA packets from (to) the TM-BUS when requested. It also sends (receives) data to (from) the memory unit. When information exchange between these buffers and memory is in progress, the slave busy bit in the SR is set; this will disable any further request from the TM-BUS. Also when these buffers are used in a TM-BUS transaction, a bit is set to prevent any processor read/write operations.

3. Cnt-0: This is a modulo-17 counter. It is used to count the number of bits received in each packet. This counter sends an overflow signal to the FSM2 in order to signal the beginning of a new packet.

4. Cnt-1: It is an up counter which can count from 0 to 255. Its function is to keep track of the number of DATA packets received on the MD line. After one TM-BUS transaction, data in this buffer can be read by the processor. The processor first reads the content of Counter-1 to determine the number of DATA packets just received. Either consecutive processor read operations or DMA operations are followed in order to move the received data to the memory unit of the MMC.

5. Cnt-2: Counter-2 is a down counter which can count from any number less than 25 down to 0. It is used to keep track of the number of DATA packets transmitted over the SD line. Before a TM-BUS transaction is initiated, either processor write operations or DMA operations are used to move data from the memory unit to the Output Buffer. A processor write operation is used to load Counter-2 with the number of data packets to be transferred.

6. Command Decoder: This is a combinational circuit. It accept 7 bit commands in the HEADER packet from inSR. The decoder is enabled only when there is no parity error and the address matches the module ID. After the HEADER packet is received, the next state of the FSM2 is determined by the output of this circuit. Some reserved standard commands are listed in [6].

7. SR (Status Register): This is an 8 bit register used to hold the status of the TM-slave. The content of this register is transmitted when an ACKNOWLEDGE packet is required or a READ STATUS command is received. Some bits in the SR can be set by signals from the Command Decoder, while the rest of them are set by functional blocks in the TM-slave. Some bits are reset when this register is read.

8. FSM2: This finite state machine carries out the TM-BUS slave protocol which is described in detail in [6]. Driven by signal lines CLOCK, CONTROL, MD and SD, where the last line is used only when a "contend for bus" command is received, the FSM2 first receives a HEADER packet and check for the destination address. If a match occurs, according to the instruction received an ACKNOWLEDGE packet is sent or DATA packets are sent and/or received.

9. Processor Interrupt Circuit: Whenever the TM-slave receives data from the TM-master, an interrupt is generated by this circuit in order to request service from the processor.

3.2.3 Processor

The functions of the processor can be classified into three categories: (1) transfer data between memory and test channels; (2) transfer data between memory and a TM-slave; (3) compare test results with good results; and (4) execute test programs. Operation speed is not crucial except for data transfer between memory and test channels, where no interrupt is allowed during a transaction. If more than 32 bits (the size of a double buffer) are required to be transferred over an ETM-BUS, data transfer between memory and test channel must be fast enough to avoid any interruption of data.

A general or special purpose 16 bit processor can be used in the MMC. It controls all other units in the MMC. Through read/write operations, the processor can access internal registers of a peripheral device, such as the TM-slave and test channels. Operations of a peripheral device can thus be controlled by a processor write to the peripheral device's control register. Data exchange between memory and a peripheral may be controlled either by processor read/write operations or by DMA operations. Any processor having the following instructions is powerful enough for the application of an MMC.

instruction	meaning
LDA R_i	Load Acc with R_i
LDA M	Load Acc with memory (M)
STA R_i	Store Acc to R_i
STA M	Store Acc to memory (M)
ADD R_i	Add R_i to Acc
AND R_i	Bitwise And R_i with Acc
CMP R_i	Compare Acc with R_i
NEG	Complement Acc
CLA	Clear Acc
BRZ R_i	Branch to (R_i) if Acc not zero
JMP R_i	Jump to (R_i)
PUSH	Push Acc onto Stack
POP	Pop Acc from Stack
NOOP	No operation
HALT	Halt the processor

The minimal architecture for a processor which is able to execute the above instruction set consists of an accumulator, four general purpose registers, an ALU, a program counter, a program status word, a stack with at least 4 words, an interrupt and DMA control circuit, and a microcode store plus microcontroller.

3.2.4 Memory

The memory unit in an MMC is composed of a RAM unit and a ROM unit. The ROM unit contains test programs to test the entire module. These programs are compiled separately before testing. Some crucial information about the chips on the module is stored here such as the number of chips to test, ordering of chips along the test bus ring, number and length of scan chains for each chip, number of random test vectors to apply for each chain, test instructions for each chip's CMC, TPG seeds and good signature for each scan chain.

The RAM unit provide scratch pad memory for test program execution. Response signatures are stored here for latter evaluation. The RAM also provides storage for the Go/NoGo status for all chips, as well as for the entire module. In addition, the RAM also provides buffer space for data transmitted to/from the TM-slave.

3.3 MMC self-test

If the MMC itself is to be testable, then each chip in the MMC must be testable and have a CMC. The MMC can then be tested like any other application chips, i.e. by some other MMC. If no MMC is available, the use of an ATE is required.

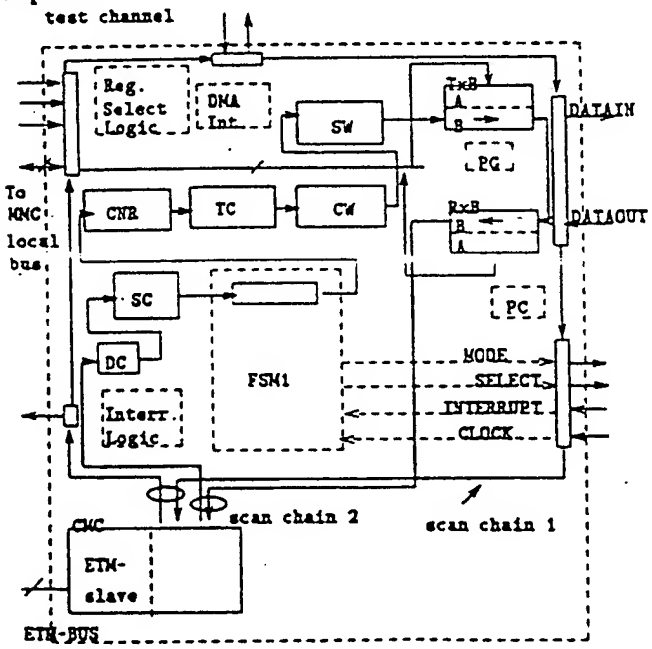


Figure 9: Testable design features for a test channel.

If the MMC is designed with an off-the-shelf processor, DMA controller and memory, only functional testing of these components is possible. However, since the TM-slaves and test channel contain a CMC, both can be tested by some MMC.

We now describe the testable design features of the test channel and TM-slave.

The testable design features of a test channel are shown in Figure 9. All combinational circuit blocks are shown in boxes with dotted lines. Register components are shown in boxes with solid lines. Some logic is associated with these registers, since some are counters and LFSRs. Normal functional connections are not shown. Instead two scan chains formed during self-test are shown. Scan chain 1 is the boundary scan chain. All I/O signals can be controlled and observed by shifting data or results along this chain. All other registers are connected to scan chain 2. The state of the test channel is controlled by shifting data along this scan chain. If a functional clock is then activated, the next state of the test channel also can be observed by shifting out the content of this chain.

During the self-test process, scan chain 1 is first loaded with test data which is held in place while the logic associated with scan chain 2 is tested. This process is repeated until a desired fault coverage is achieved.

The TM-slave is made testable by using three scan chains. Scan chain 1 contains all boundary scan registers of the TM-slave. All primary I/O signals can be controlled and observed through this chain.

Scan chain 2 contains all other internal registers except Input Buffer and Output Buffer. The state of the TM-slave can be controlled by shifting data into this chain. After a functional clock is activated, the next state of the TM-slave can be observed by shifting out results along this chain.

Scan chain 3 contains only Input Buffer and Output Buffer. These two buffers are implemented with RAMs, and BIST capabilities can be initiated via scan chain 2. If no BIST capability is provided, then a long sequence of set-scan operations is required to test both buffers.

4 Data Transfer Rate Analysis

The data transfer rate requirements between memory and a test channel is calculated in the following. The result shows that unless a RISC-type processor is used, in which only one clock cycle is required for memory access, a DMA controller is required to ensure fast enough data transfer in order to prevent any data interruption on the ETM-BUS.

Suppose DTD type information is transmitted to one or more chips, and the internal registers of the controlling test channel have been initialized. Recall that both the TxB

and RxB are designed with double buffers. Suppose a test vector is longer than 32 bits, and both buffers of TxB are filled before the transfer is started. After 16 (ETM-BUS) clock cycles, one buffer of TxB is empty and one buffer of RxB is filled with status information. To prevent any interruption on the DATAIN line, the other buffer of TxB must be filled in 16 clock cycles. During that time, the filled buffer of RxB must also be read out to prevent interruption on the DATAOUT line. In short, within 16 clock cycles, 16 bits of data are sent to the test channel, and another 16 bits of data are received from it.

To simplify the analysis, assume that the same clock drives both the processor and the ETM-BUS, and a non RISC-type processor is used. Also suppose that only one test channel is controlled by the processor, so it can poll the test channel and no interrupt is required to get service. Requesting service through an interrupt takes more clock cycles than direct polling by the processor. The analysis given below shows that even if the processor needs to poll only one test channel, it is not fast enough for this particular application.

In the worst case two instructions, namely a *LDA M* to read a channel's status followed by a *BRZ Ri* to check the status are required for the polling process. Then two *LDA M* and two *STA M* are performed. These instructions are used to transfer a word from memory into Acc (*LDA M*) and write from Acc to TxB (*STA M*), and to read from the RxB (*LDA M*) and write to memory (*STA M*). Here an internal register of a test channel is accessed by the processor as a special memory location (assigned for I/O). Thus 6 instructions, which including 5 memory accesses and 1 conditional branch need to be executed within 16 clock cycles. Assume the conditional branch can be executed in 2 clock cycles, then 14 clock cycles are left for 5 memory access. Thus for a processor to be able to control such a process, a memory access instruction should not take more than 2 clock cycles to execute. Most general purpose processors can not meet this requirement. If several test channels are operated at the same time the situation is even worse.

One solution to this problem is to increase the size of the buffers for TxB and RxB so that all the data which must be sent in one transmission can be stored in the buffers. The problem is that a large memory may be required.

Another solution is to use a DMA controller, such as the Intel 8237A, which provides 4 DMA channels. The processor can be in the idle state when a transfer is in progress. Two DMA channels are used to control a test channel. One for transferring data from memory to TxB, the other for RxB to memory. Since there is only one internal data bus per test channel, only unidirection transfers can occur at a time. But two DMA transfers must be done in 16 clock cycles. Both DMA channels are programmed in the Single Transfer Mode. The 8237A is 8 bit in width. However, since the operation performed is memory to I/O (TxB) or vice verse, and no data is coming in or going out of the DMA controller, it can be used to control 16 bit data transfers, which is required in this application.

A DMA memory to I/O operation takes 4 clock cycles, with another cycle required for recognizing a DMA request. This adds up to 5 clock cycles to move 16 bits of data from memory to the TxB. The second transfer (from the RxB to memory) requires an additional 5 clock cycles. So 10 clock cycles are required for a transfer to and from memory. This data transfer rate is sufficient to keep one test channel busy.

5 Conclusions

We have described an MMC design suitable for controlling the self-test process of a module containing testable chips. The proposed test channel design is compatible with both Boundary Scan and ETM-BUS protocols. The test channel prevents the processor from dealing with detailed bus control sequences. Test execution sequences for chips can be controlled in terms of processor read/write operations, which greatly simplifies the development of test programs.

The MMC architecture is expandable, i.e. more test channels can be added to increase the parallelism of testing. Additional DMA controllers must be added if the number of test channels employed in an MMC exceeds the maximum limit in which a DMA controller can control.

Four or more clocks may be applied to an MMC, viz. TCK for CMC, FCK1 for TM-slave, FCK2 for each test channel connected to an ETM-BUS, and FCK3 for operation among processor and other peripheral devices. Synchronization problems will occur in a test channel, where both FCK2 and FCK3 access the same component, such as TxB. The same problem occurs in the TM-slave, where FCK1 and FCK3 access both the Input Buffer and Output Buffer. Techniques to solve this problem can be found in [20,21]. In the design presented here, we assume that the same clock source is used for all these clocks, thus avoiding the clock synchronization problem.

Test programs for an MMC are easy to write since most chips employ similar types of test hardware, such as scan chains. It is our believe that it is possible to automatically synthesize test program for an MMC.

The proposed MMC is designed to be part of a HTM system. It is assumed that each module contains an MMC, which under the request of a SuMP can test all chips on the

module and report back test results. However, it is possible to build an MMC as a portable stand-alone unit. In this case the TM-slave can be removed without affecting the function of an MMC since it is not controlled by a SuMP. Instead, a control panel is required. A stand-alone MMC can test any module having an ETM-BUS. The module's built-in MMC is tested first through its ETM-slave. Application chips on the module can be tested either by the built-in MMC or by a stand-alone MMC. For the latter, the built-in MMC on that module must be disabled to allow the stand-alone MMC to control the module's ETM-BUS.

An operator can start the test process via the control panel. Test programs stored in the ROM then take over control. After all chips have been tested, test results are shown on the control panel to indicate the Go/NoGo status of the module under test.

References

[1] F. Beenker, et al., *Macro Testing: Unifying IC and Board Test*, IEEE Design & Test of Computers, December 1986, pp 26-32.

[2] F. Beenker, *Systematic and Structured Methods for Digital Board Testing*, VLSI System Design, January 1987, pp 50-58.

[3] C. Maunder and F. Beenker, *BOUNDARY-SCAN: A Framework for Structured Design-For-Test*, Proc. Int'l Test Conference, 1987, pp 714-723.

[4] D. van de Lagemaat and H. Bleeker, *Testing a Board with Boundary Scan*, Proc. Int'l Test Conference, 1987, pp 724-729.

[5] IBM, Honeywell and TRW, *VHSIC Phase 2 INTEROPERABILITY STANDARDS*, ETM-BUS specification, December 1986.

[6] IBM, Honeywell and TRW, *VHSIC Phase 2 INTEROPERATABILITY STANDARDS*, TM-BUS specification, December 1986.

[7] Technical Subcommittee of JTAG, *Boundary-Scan Architecture Standard Proposal*, Version 2.0, March 1988.

[8] M.A. Breuer, *On-Chip Controller Design for Built-In-Test*, Technical Report CRI-88-04, Dept. of EE-Systems, University of Southern California, December 1985.

[9] M.A. Breuer, R. Gupta, and J.C. Lien, *Concurrent Control of Multiple BIT Structures*, Proc. Int'l Test Conference (this issue), 1988.

[10] C.L. Hudson, Jr. and G.D. Peterson, *Parallel Self-Test with Pseudo-Random Test Patterns*, Proc, Int'l Test Conference, 1987, pp 954-963.

[11] LaNae Avra, *A VHSIC ETM-BUS Compatible Test and Maintenance Interface*, Proc. Int'l Test Conference, 1987, pp 964-971.

[12] J.J. LeBlanc, *LOCST: A Built-In Self-Test Technique*, IEEE Design & Test of Computers, November 1984, pp 45-52.

[13] IEEE P1149 T-BUS Standardization Committee, *P1149 Draft 5*, News Letter from P1149 Co-Chairs, April 1988.

[14] E.J. McCluskey, *Built-In Self-Test Structures*, IEEE Design & Test of Computers, April 1985, pp 29-36.

[15] E.J. McCluskey, *Built-In Self-Test Techniques*, IEEE Design & Test of Computers, April 1985, pp 21-28.

[16] E.B. Eichelberger and T.W. Williams, *A Logic Design Structure For LSI Testability*, Proc. 14th Design Automation Conference, June 1977, pp 462-467.

[17] B. Konemann, J. Mucha and G. Zwiehoff, *Built-In Logic Block Observation Techniques*, Proc. Int'l Test Conference, 1979, pp 37-41.

[18] M.A. Breuer and J.C. Lien, *A Methodology for the Design of Hierarchically Testable and Maintainable Digital Systems*, to appear in Proc. 8th Digital Avionics Systems Conference, October 1988.

[19] J.E. Haedtke and W.R. Olson, *Multilevel Self-Test for the Factory and Field*, Proc. Annual Reliability and Maintainability Symposium, 1987, pp 274-279.

[20] S.Y. Kung, S.C. Lo, S.N. Jean and J.N. Hwang, *Wavefront Array Processors — Concept to Implementation*, IEEE Computer, July 1987, pp 18-33.

[21] G. Borriello and R. H. Katz, *Synthesis and Optimization of Interface Transducer Logic*, Proc. Int'l Conference on Computer-Aided Design, 1987, pp 274-277.